Package ‘Boruta’

January 27, 2017

Title Wrapper Algorithm for All Relevant Feature Selection
Version 5.2.0
Depends ranger
Suggests mlbench, rFerns, randomForest
Description An all relevant feature selection wrapper algorithm. It finds relevant features by comparing original attributes’ importance with importance achievable at random, estimated using their permuted copies.

BugReports https://github.com/mbq/Boruta/issues
License GPL (>= 2)
URL https://m2.icm.edu.pl/boruta/
RoxygenNote 5.0.1
NeedsCompilation no
Author Miron Bartosz Kursa [aut, cre], Witold Remigiusz Rudnicki [aut]
Maintainer Miron Bartosz Kursa <M.Kursa@icm.edu.pl>
Repository CRAN
Date/Publication 2017-01-27 10:17:32

R topics documented:

attStats ......................................................... 2
Boruta ......................................................... 3
getConfirmedFormula ........................................ 6
getImpFerns .................................................. 6
getImpLegacyRf ............................................. 7
getImpRf ...................................................... 8
getSelectedAttributes .................................... 9
plot.Boruta .................................................. 10
plotImpHistory ............................................. 11
print.Boruta ................................................ 12
TentativeRoughFix ........................................ 13
attStats

Description

attStats shows a summary of a Boruta run in an attribute-centred way. It produces a data frame containing some importance stats as well as the number of hits that attribute scored and the decision it was given.

Usage

attStats(x)

Arguments

x an object of a class Boruta, from which attribute stats should be extracted.

Value

A data frame containing, for each attribute that was originally in information system, mean, median, maximal and minimal importance, number of hits normalised to number of importance source runs performed and the decision copied from finalDecision.

Note

When using a Boruta object generated by a TentativeRoughFix, the resulting data frame will consist a rough-fixed decision.

x has to be made with holdHistory set to TRUE for this code to run.

Author(s)

Miron B. Kursa

Examples

## Not run:
library(mlbench); data(Sonar);
#Takes some time, so be patient
Boruta(Class~.,data=Sonar,doTrace=2)->Bor.son;
print(Bor.son);
stats<-attStats(Bor.son);
pnormHits=meanImp,col=stats$decision,data=stats);

## End(Not run)
Description

Boruta is an all relevant feature selection wrapper algorithm, capable of working with any classification method that output variable importance measure (VIM); by default, Boruta uses Random Forest. The method performs a top-down search for relevant features by comparing original attributes' importance with importance achievable at random, estimated using their permuted copies, and progressively eliminating irrelevant features to stabilise that test.

Usage

```r
Boruta(x, ...)

## Default S3 method:
Boruta(x, y, pValue = 0.01, mcAdj = TRUE, maxRuns = 100,
       doTrace = 0, holdHistory = TRUE, getImp = getImpRf, ...)

## S3 method for class 'formula'
Boruta(formula, data = .GlobalEnv, ...)
```

Arguments

- **x**
  - data frame of predictors.
- **...**
  - additional parameters passed to `getImp`.
- **y**
  - response vector; factor for classification, numeric vector for regression, `Surv` object for survival (supports depends on importance adapter capabilities).
- **pValue**
  - confidence level. Default value should be used.
- **mcAdj**
  - if set to TRUE, a multiple comparisons adjustment using the Bonferroni method will be applied. Default value should be used; older (1.x and 2.x) versions of Boruta were effectively using FALSE.
- **maxRuns**
  - maximal number of importance source runs. You may increase it to resolve attributes left Tentative.
- **doTrace**
  - verbosity level. 0 means no tracing, 1 means reporting decision about each attribute as soon as it is justified, 2 means same as 1, plus reporting each importance source run.
- **holdHistory**
  - if set to TRUE, the full history of importance is stored and returned as the `ImpHistory` element of the result. Can be used to decrease a memory footprint of Boruta in case this side data is not used, especially when the number of attributes is huge; yet it disables plotting of such made Boruta objects and the use of the `TentativeRoughFix` function.
Boruta

getImp  function used to obtain attribute importance. The default is getImpRfZ, which runs random forest from the ranger package and gathers Z-scores of mean decrease accuracy measure. It should return a numeric vector of a size identical to the number of columns of its first argument, containing importance measure of respective attributes. Any order-preserving transformation of this measure will yield the same result. It is assumed that more important attributes get higher importance. \(+\infty\) are accepted, NaNs and NAs are treated as 0s, with a warning.

formula  alternatively, formula describing model to be analysed.

data  in which to interpret formula.

Details

Boruta iteratively compares importances of attributes with importances of shadow attributes, created by shuffling original ones. Attributes that have significantly worst importance than shadow ones are being consecutively dropped. On the other hand, attributes that are significantly better than shadows are admitted to Confirmed. Shadows are re-created in each iteration. Algorithm stops when only Confirmed attributes are left, or when it reaches maxRuns importance source runs. If the second scenario occurs, some attributes may be left without a decision. They are claimed Tentative. You may try to extend maxRuns or lower pValue to clarify them, but in some cases their importances do fluctuate too much for Boruta to converge. Instead, you can use TentativeRoughFix function, which will perform other, weaker test to make a final decision, or simply treat them as undecided in further analysis.

Value

An object of class Boruta, which is a list with the following components:

finalDecision  a factor of three value: Confirmed, Rejected or Tentative, containing final result of feature selection.

ImpHistory  a data frame of importances of attributes gathered in each importance source run. Beside predictors’ importances, it contains maximal, mean and minimal importance of shadow attributes in each run. Rejected attributes get \(-\infty\) importance. Set to NULL if holdHistory was given FALSE.

timeTaken  time taken by the computation.

impSource  string describing the source of importance, equal to a comment attribute of the getImp argument.

call  the original call of the Boruta function.

Note

Version 5.0 and 2.0 change some name conventions and thus may be incompatible with scripts written for earlier Boruta versions. Solutions of most problems of this kind should boil down to change of ZScoreHistory to ImpHistory in script source or Boruta object structure.

Author(s)

Miron B. Kursa, based on the idea & original code by Witold R. Rudnicki.
References


Examples

```r
set.seed(777);
# Add some nonsense attributes to iris dataset by shuffling original attributes
iris.extended <- data.frame(iris, apply(iris[,-5], 2, sample));
names(iris.extended)[6:9] <- paste("Nonsense",1:4,sep="");
# Run Boruta on this data
Boruta(Species~., data=iris.extended, doTrace=2) -> Boruta.iris.extended
# Nonsense attributes should be rejected
print(Boruta.iris.extended);

# Boruta using rFerns' importance
Boruta(Species~., data=iris.extended, getImp = getImpFerns) -> Boruta.ferns.irisE
print(Boruta.ferns.irisE);

## Not run:
## Boruta on the HouseVotes84 data from mlbench
library(mlbench); data(HouseVotes84);
na.omit(HouseVotes84) -> hvo;
# Takes some time, so be patient
Boruta(Class~., data=hvo, doTrace=2) -> Bor.hvo;
print(Bor.hvo);
plot(Bor.hvo);
plotImpHistory(Bor.hvo);

## End(Not run)
## Not run:
## Boruta on the Ozone data from mlbench
library(mlbench); data(Ozone);
library(randomForest);
aa.omit(Ozone) -> ozo;
Boruta(V4~., data=ozo, doTrace=2) -> Bor.ozo;
cat('Random forest run on all attributes:\n');
print(randomForest(V4~., data=ozo));
cat('Random forest run only on confirmed attributes:\n');
print(randomForest(ozo[,getSelectedAttributes(Bor.ozo)], ozo$V4));

## End(Not run)
## Not run:
## Boruta on the Sonar data from mlbench
library(mlbench); data(Sonar);
# Takes some time, so be patient
Boruta(Class~., data=Sonar, doTrace=2) -> Bor.son;
print(Bor.son);
# Shows important bands
plot(Bor.son, sort=FALSE);

## End(Not run)
```
getConfirmedFormula  \hspace{1cm} Export Boruta result as a formula

Description

Functions which convert the Boruta selection into a formula, so that it could be passed further to other functions.

Usage

getConfirmedFormula(x)

getNonRejectedFormula(x)

Arguments

x  \hspace{1cm} an object of a class Boruta, made using a formula interface.

Value

Formula, corresponding to the Boruta results. getConfirmedFormula returns only Confirmed attributes, getNonRejectedFormula also adds Tentative ones.

Note

This operation is possible only when Boruta selection was invoked using a formula interface.

getImpFerns  \hspace{1cm} Random Ferns importance

Description

This function is intended to be given to a getImp argument of Boruta function to be called by the Boruta algorithm as an importance source.

Usage

getImpFerns(x, y, ...)

Arguments

x  \hspace{1cm} data frame of predictors including shadows.

y  \hspace{1cm} response vector.

...  \hspace{1cm} parameters passed to the underlying rFerns call; they are relayed from ... of Boruta.
Note

Random Ferns importance calculation should be much faster than using Random Forest; however, one must first optimize the value of the depth parameter and it is quite likely that the number of ferns in the ensemble required for the importance to converge will be higher than the number of trees in case of Random Forest.

getImpLegacyRf  randomForest importance adapters

Description

Those function is intended to be given to a getImp argument of Boruta function to be called by the Boruta algorithm as an importance source. getImpLegacyRfZ generates default, normalized permutation importance, getImpLegacyRfRaw raw permutation importance, finally getImpLegacyRfGini generates Gini index importance, all using randomForest as a Random Forest algorithm implementation.

Usage

getImpLegacyRfZ(x, y, ...)  
getImpLegacyRfRaw(x, y, ...)  
getImpLegacyRfGini(x, y, ...)

Arguments

x  data frame of predictors including shadows.  
y  response vector.  
...  parameters passed to the underlying randomForest call; they are relayed from ... of Boruta.

Note

The getImpLegacyRfZ function was a default importance source in Boruta versions prior to 5.0; since then ranger Random Forest implementation is used instead of randomForest, for speed, memory conservation and an ability to utilise multithreading. Both importance sources should generally lead to the same results, yet there are differences.

Most notably, ranger by default treats factor attributes as ordered (and works very slow if instructed otherwise with respect unordered factors=TRUE); on the other hand it lifts 32 levels limit specific to randomForest. To this end, Boruta decision for factor attributes may be different.

Random Forest methods has two main parameters, number of attributes tried at each split and the number of trees in the forest; first one is called mtry in both implementations, but the second ntree in randomForest and num.trees in ranger. To this end, to maintain compatibility, getImpRf* functions still accept ntree parameter relaying it into num.trees. Still, both parameters take the
same defaults in both implementations (square root of the number all attributes and 500 respectively).
Moreover, ranger brings some addition capabilities to Boruta, like analysis of survival problems or sticky variables which are always considered on splits.
Finally, the results for the same PRNG seed will be different.

Examples

```r
set.seed(777);
#Add some nonsense attributes to iris dataset by shuffling original attributes
iris.extended <- data.frame(iris, apply(iris[, -5], 2, sample));
names(iris.extended)[6:9] <- paste("Nonsense", 1:4, sep="");
#Run Boruta on this data
Boruta(Species~., getImp=getImpLegacyRfZ,
data=iris.extended, doTrace=2) -> Boruta.iris.extended
#Nonsense attributes should be rejected
print(Boruta.iris.extended);
```

### `getImpRf`

#### ranger Random Forest importance adapters

**Description**

Those function is intended to be given to a getImp argument of Boruta function to be called by the Boruta algorithm as an importance source. `getImpRfZ` generates default, normalized permutation importance, `getImpRfRaw` raw permutation importance, finally `getImpRfGini` generates Gini index importance.

**Usage**

```r
getImpRfZ(x, y, ntree = 500, num.trees = ntree, ...)
getImpRfGini(x, y, ntree = 500, num.trees = ntree, ...)
getImpRfRaw(x, y, ntree = 500, num.trees = ntree, ...)
```

**Arguments**

- `x` data frame of predictors including shadows.
- `y` response vector.
- `ntree` Number of trees in the forest; copied into ranger’s native num.trees, put to retain transparent compatibility with randomForest.
- `num.trees` Number of trees in the forest, as according to ranger’s nomenclature. If not given, set to ntree value. If both are given, num.trees takes precedence.
- `...` parameters passed to the underlying ranger call; they are relayed from ... of Boruta.
**getSelectedAttributes**

**Note**

Prior to Boruta 5.0, `getImpLegacyRfZ` function was a default importance source in Boruta; see `getImpLegacyRf` for more details.

---

**getSelectedAttributes**  *Extract names of the selected attributes*

**Description**

`getSelectedAttributes` returns a vector of names of attributes selected during a Boruta run.

**Usage**

```r
getSelectedAttributes(x, withTentative = FALSE)
```

**Arguments**

- **x**: an object of a class Boruta, from which relevant attributes names should be extracted.
- **withTentative**: if set to TRUE, Tentative attributes will be also returned.

**Value**

A character vector with names of the relevant attributes.

**Author(s)**

Miron B. Kursa

**Examples**

```r
## Not run:
data(iris);
#Takes some time, so be patient
Boruta(Species~., data=iris, doTrace=2)->Bor.iris;
print(Bor.iris);
print(getSelectedAttributes(Bor.iris));

## End(Not run)
```
plot.Boruta

Plot Boruta object

Description

Default plot method for Boruta objects, showing boxplots of attribute importances over run.

Usage

```r
## S3 method for class 'Boruta'
plot(x, colCode = c("green", "yellow", "red", "blue"),
     sort = TRUE, whichShadow = c(TRUE, TRUE, TRUE), col = NULL,
     xlab = "Attributes", ylab = "Importance", ...)
```

Arguments

- `x`: an object of a class Boruta.
- `colCode`: a vector containing colour codes for attribute decisions, respectively Confirmed, Tentative, Rejected and shadow.
- `sort`: controls whether boxplots should be ordered, or left in original order.
- `whichShadow`: a logical vector controlling which shadows should be drawn; switches respectively max shadow, mean shadow and min shadow.
- `xlab`: X axis label that will be passed to `boxplot`.
- `ylab`: Y axis label that will be passed to `boxplot`.
- `...`: additional graphical parameter that will be passed to `boxplot`.

Value

Invisible copy of `x`.

Note

If `col` is given and `sort` is TRUE, the `col` will be permuted, so that its order corresponds to attribute order in `ImpHistory`.

This function will throw an error when `x` lacks importance history, i.e., was made with `holdHistory` set to FALSE.

Author(s)

Miron B. Kursa
Examples

```r
## Not run:
library(mlbench); data(HouseVotes84);
na.omit(HouseVotes84) -> hvo;
# Takes some time, so be patient
Boruta(Class~., data=hvo, doTrace=2) -> Bor.hvo;
print(Bor.hvo);
plot(Bor.hvo);

## End(Not run)
```

---

**plotImpHistory**  
*Plot Boruta object as importance history*

**Description**

Alternative plot method for Boruta objects, showing matplot of attribute importances over run.

**Usage**

```r
plotImpHistory(x, colCode = c("green", "yellow", "red", "blue"),
              col = NULL,
              type = "l", lty = 1, pch = 0, xlab = "Classifier run",
              ylab = "Importance", ...)
```

**Arguments**

- `x`  
an object of a class Boruta.

- `colCode`  
a vector containing colour codes for attribute decisions, respectively Confirmed, Tentative, Rejected and shadow.

- `col`  
standard col attribute, passed to `matplot`. If given, suppresses effects of `colCode`.

- `type`  
Plot type that will be passed to `matplot`.

- `lty`  
Line type that will be passed to `matplot`.

- `pch`  
Point mark type that will be passed to `matplot`.

- `xlab`  
X axis label that will be passed to `matplot`.

- `ylab`  
Y axis label that will be passed to `matplot`.

- `...`  
additional graphical parameter that will be passed to `matplot`.

**Value**

Invisible copy of `x`.

**Note**

This function will throw an error when `x` lacks importance history, i.e., was made with `holdHistory` set to `FALSE`. 
print.Boruta

Author(s)
Miron B. Kursa

Examples

```r
## Not run:
library(mlbench); data(Sonar);
# Takes some time, so be patient
Boruta(Class~.,data=Sonar,doTrace=2)->Bor.son;
print(Bor.son);
plotImpHistory(Bor.son);

## End(Not run)
```

print.Boruta  

Print Boruta object

Description
Print method for the Boruta objects.

Usage

```r
## S3 method for class 'Boruta'
print(x, ...)
```

Arguments

- `x` an object of a class Boruta.
- `...` additional arguments passed to `print`.

Value
Invisible copy of `x`.

Author(s)
Miron B. Kursa
TentativeRoughFix

Rough fix of Tentative attributes

Description

In some circumstances (too short Boruta run, unfortunate mixing of shadow attributes, tricky dataset...), Boruta can leave some attributes Tentative. TentativeRoughFix performs a simplified, weaker test for judging such attributes.

Usage

TentativeRoughFix(x, averageOver = Inf)

Arguments

x an object of a class Boruta.
averageOver Either number of last importance source runs to average over or Inf for averaging over the whole Boruta run.

Details

Function claims as Confirmed those attributes that have median importance higher than the median importance of maximal shadow attribute, and the rest as Rejected. Depending of the user choice, medians for the test are count over last round, all rounds or N last importance source runs.

Value

A Boruta class object with modified finalDecision element. Such object has few additional elements:
originalDecision Original finalDecision.
averageOver Copy of averageOver parameter.

Note

This function should be used only when strict decision is highly desired, because this test is much weaker than Boruta and can lower the confidence of the final result.
x has to be made with holdHistory set to TRUE for this code to run.

Author(s)

Miron B. Kursa
Index

attStats, 2
Boruta, 3, 6–8
boxplot, 10

getConfirmedFormula, 6
getImpFerns, 6
getImpLegacyRf, 7, 9
getImpLegacyRfGini (getImpLegacyRf), 7
getImpLegacyRfRaw (getImpLegacyRf), 7
getImpLegacyRfZ (getImpLegacyRf), 7
getImpRf, 8
getImpRfGini (getImpRf), 8
getImpRfRaw (getImpRf), 8
getImpRfZ (getImpRf), 8
getLegacyImpRfRaw (getImpLegacyRf), 7
getNonRejectedFormula
  (getConfirmedFormula), 6
getSelectedAttributes, 9

matplot, 11

plot.Boruta, 10
plotImpHistory, 11
print, 12
print.Boruta, 12

randomForest, 7
ranger, 7, 8
rFerns, 6

TentativeRoughFix, 2–4, 13